Computer-Aided Design and Optimization of High-Performance Vacuum Electronic Devices

SBIR Phase 1
Final Report

Contract/Purchase Order Number: N00014-04-M-0169

November 24, 2004

John F. DeFord, Ben Held, and Liya Chernyakova
Simulation Technology & Applied Research, Inc.
11520 N. Port Washington Rd., Suite 101B
Mequon, WI 53092

P: 1-262-240-0291 x102
F: 1-262-240-0294
E: john.deford@staarinc.com

John Petillo
Scientific Applications International Corporation
Suite 130, 20 Burlington Mall Rd.
Burlington, MA 01813

P: 1-781-221-7615
F: 1-781-270-0063
E: jpetillo@bos.saic.com

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
24 NOV 2004	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Computer-Aided Design and Optimization of High-Performance Vacuum Electronic Devices	**N00014-04-M-0169**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Scientific Applications International Corp., Suite 130, 20 Burlington Mall Rd., Burlington, MA 01813**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
Per John F. DeFord this report is unclassified/unlimited, The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**9**	

Overview

The layout of the system that we developed during the Phase 1 project is shown in Fig. 1. The primary components of the system are Analyst[1], Mathematica[2], and MICHELLE[3]. Analyst is a commercial electromagnetic analysis tool and acts as the integration platform. It supports the other codes and controls the overall process. Mathematica is a commercial symbolic mathematics package that performs the optimization (under control of Analyst). MICHELLE is a 3D gun code developed by SAIC that was used to in our study.

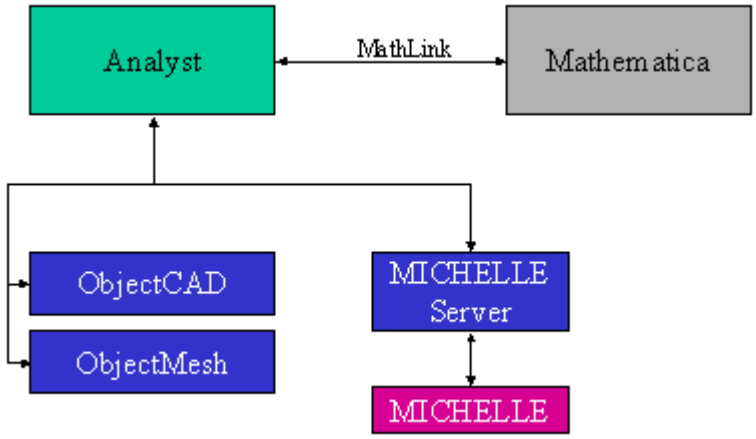

Fig. 1. System architecture. Analyst uses its ObjectCAD and ObjectMesh servers to create geometry and generate finite element meshes, and it can also generate input files for MICHELLE, run the solver, and process output files.

Mathematica is used to simplify the experimentation with various optimization strategies. The actual optimization algorithms are coded in Mathematica's symbolic mathematics language, greatly simplifying the process of testing various methods. Moreover, several common optimization methods are already available in Mathematica as native functions: the statistical methods differential evolution and simulated annealing, and the Nelder-Mead simplex method. Testing these methods does not require any special coding.

In the most recent period of the Phase 1 project we have extended the MICHELLE interface in Analyst to allow the use of geometric parameters in an optimization. The restriction on the number of parameters was also removed, and optimizations with up to 14 parameters were performed.

The following system functionality has been demonstrated:

- Optimization over arbitrary number of parameters.
- Inclusion of secondaries in the analysis.
- Use of adaptive mesh refinement (AMR) in the analysis.

---

[1] Analyst is a commercial finite-element analysis package. See www.staarinc.com for more details.
[2] Mathematica is a commercial symbolic mathematics package. See www.wolframresearch.com for more details.
[3] J. Petillo, et al., "The MICHELLE three-dimensional electron gun and collector modeling tool: theory and design," IEEE Trans. Plasma Science, 30, June, 2002, pp. 1238-1264.

In tests on an idealized collector geometry we were able to show improvements in efficiency over an initial design of as much as 16% with differential evolution and the Nelder-Mead method using various strategies.

Enhancements to the MICHELLE Interface in Analyst

An interface in Analyst to the MICHELLE solver was created early in the project. Our most recent work has extended this interface to allow the specification of geometric parameters as optimization targets.

After a model has been created in Analyst, an optimization is performed using the optimization "wizard", which is a sequence of panels that help set up the process (Fig. 2). The goal function is defined in Mathematica, and it is nominally a function of both the parameters and the result metrics (collector efficiency in this case). The link to Mathematica is discussed in more detail in the next section.

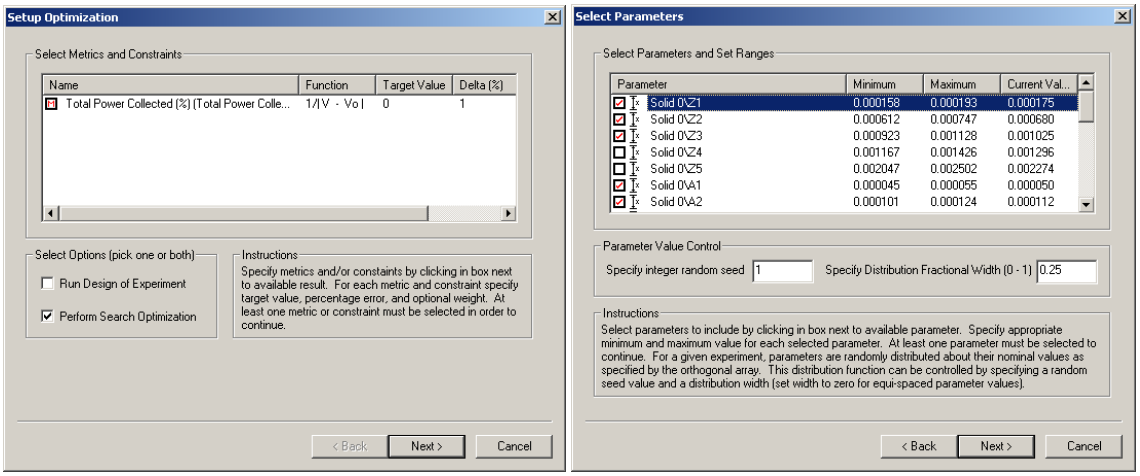

Fig. 2. First two panels of the optimization "wizard" within Analyst. The first panel is used to define what is to be optimized, and the second panel is used to pick which parameters to include in the optimization, and to define initial values and valid ranges.

For the purposes of this work, the goal function was defined as the square of the collector efficiency, and Mathematica was instructed to find the maximum of the goal function.

Optimization Link to Mathematica

The current optimization architecture employed within Analyst makes use of Mathematica to control the optimization process. Via its MathLink interface, Analyst provides Mathematica with information about the number of parameters, parameter ranges, algorithm selection, etc.

Mathematica calls a generic Analyst interface function every time it wishes to evaluate a parameter vector. This problem independent function updates the geometry and boundary conditions to reflect the requested parameter vector, runs the analysis, and returns the selected results (e.g. efficiency). A metric function (defined by the user in the Mathematica language) combines the results (if there is more than one) and returns an appropriate metric value to the optimization routine. This function can be customized to include weighting particular results or imposing specialized constraints.

A table of all of the analyses that were performed is available to the user as the optimization is running. Mathematica determines when to stop the optimization based on user-specified convergence criteria.

Optimization Methods

Three different optimization methods have been considered so far: simulated annealing (SA), differential evolution (DE), and the Nelder-Mead (NM) method. Brief descriptions of these methods are given in the following paragraphs. Note that these methods can be used to find minima or maxima, but for simplicity we refer only to minima in the discussion.

Simulated Annealing[4]

In this method an initial random parameter vector is generated that satisfies the parameter constraints $r_{\min}^{(i)} \le r_0^{(i)} \le r_{\max}^{(i)}$, and the objective function is tested at this point. Subsequent parameter vectors are chosen using an expression of the form

$$\mathbf{r}_n = \mathbf{r}_{n-1} + \frac{T}{T_0}\left[a(\mathbf{r}_{\max} - \mathbf{r}_{n-1}) + b(\mathbf{r}_{n-1} - \mathbf{r}_{\min})\right]$$

where T is the current "temperature" (by analogy with actual annealing), T_0 is an initial temperature, and a and b are bounded random variables. The objective function is tested for each new parameter vector, and the new parameter vector is kept if the objective function is reduced. The parameter vector may also be kept if the objective function increases, with a probability given by the Boltzmann factor:

$$p = e^{-\Delta f / T}$$

where Δf is the normalized change in the objective function (this feature enables the process to "escape" from local minima).

SA proceeds by generating and evaluating new parameter vectors at a constant "temperature" until adequate statistics are obtained, then the temperature is reduced and the process is repeated. The efficiency and result of the optimization can be quite dependent on the initial temperature and the "cooling schedule", that is, the way the temperature is varied during the process.

SA is used in circuit design and other endeavors where objective function evaluations are relatively inexpensive. Our experience with it in collector optimization is that it requires too many analyses to be useful in most cases.

Differential Evolution[5]

This method comes from a class algorithms based upon evolutionary principles. To start the process, an initial "population" of random vectors $\{\mathbf{p}_{k,0}\}$ is created that all satisfy the parameter constraints. At each iteration (called a "generation") of the process, new vectors are obtained from the previous set using the following concepts:

[4] S. Kirkpatrick, et al., "Optimization by simulated annealing," *Science*, **220**, May, 1983, pp. 671-680.
[5] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, **11**, 1997, pp. 341-359.

- *Mutation*. A new vector is formed via a combination of existing vectors of the form
$$\mathbf{v}_{i,G+1} = \mathbf{p}_{i,G} + \alpha\left(\mathbf{p}_{k,G} - \mathbf{p}_{l,G}\right)$$
- *Recombination*. A candidate "child" vector is formed by taking some (randomly selected) parameter values directly from the parent $\mathbf{p}_G$, and the rest from the differential combination vector $\mathbf{v}_G$, i.e.,
$$\mathbf{u}_{i,G+1} = \begin{cases} \mathbf{v}_{i,G}, & i \in S \\ \mathbf{p}_{i,G}, & i \notin S \end{cases}$$
- *Selection*. A parent vector is replaced with a child vector if the objective function is reduced. Otherwise, additional children are created and tested until either one is found that reduces the objective function or some maximum number of offspring is reached. If no child is more "fit" than the parent, the parent passes to the new generation (if they are not eliminated by the aging criterion below).
- *Aging*. A vector can only "survive" for a limited number of generations, regardless of its "fitness".

As DE proceeds, the population becomes increasingly homogeneous, until at convergence all of the vectors are the same. The process is dependent on the size of the population that is maintained, on the number of children that can be generated at each step, and other parameters. As with SA, DE can escape from local minima. Moreover, in our tests it generally was more efficient than SA.

Nelder-Mead[6]

This is a direct search method in which a simplex of dimension n+1 is updated (for an n-dimensional parameter vector) at each step. The volume enclosed by the simplex generally reduces until it encloses a minimum of the objective function.

A step in the process begins with stored values of the simplex vertices (parameter vectors) and the associated objective function values. The parameter vector corresponding to the center of the simplex is formed and tested. If the objective function at that location is less than at any of the simplex vertices, then a new simplex is formed by replacing the vertex with the highest objective function value with the new vector. If the objective function at the center is higher than at one or more of the vertices, a different point is selected and tested using a set of simple rules.

There is relatively little theory on the performance of this method. It may not converge to an extremum, or may not converge at all. Nevertheless, it is very popular and is used in a variety of application areas because it generally works quite well, and because it is conceptually and programmatically simple. We found it to be, by far, the most efficient of the methods we tested.

Optimization of 5-Stage Collector

Our design system was applied to the optimization of a five-stage collector suggested by Boeing[7] (see Figs. 3 and 4). This structure does not correspond to an actual device, so no

[6] J. Lagarias, et al., "Convergence properties of the Nelder-Mead simplex method in low dimensions," *SIAM J. Optim.*, **9**, 1998, pp. 112-147.

measured data are available for comparison. However, it serves to illustrate the principle characteristics of the optimization procedure. The electron beam input to the collector is from a traveling-wave tube with circuit efficiency of 83.9%.

An initial set of optimizations was performed to test the system, and the results from these runs are summarized in Table 1. For these runs we used a relatively coarse mesh (11K tetrahedrons), no adaptive mesh refinement, and no secondaries. There were a total of 14 parameters the could be varied, and we took two approaches to the optimization:

- *Staged*. In this approach we did a sequence of three-parameter optimizations. Starting with the first stage we optimized over its parameters (V_1, A_1, Z_1), then fixed these values at the optimum point and repeated the process for the next stage until all stages were completed (the last stage only has two parameters). The initial values and results after each stage are shown in the first 6 columns of Table 1. Nelder-Mead was used for these optimizations.
- *All parameters at once*. Here we did a single 14-parameter optimization. The result of this exercise is shown in the last column of Table 1. Nelder-Mead was used for this optimization.

We also used differential evolution to optimize do the first stage of the optimization (varying V_1, A_1, Z_1). Although this process involved more than 600 analyses (as compared to 57 for NM), it achieved an efficiency 2% higher than was obtained with Nelder-Mead. For situations in which small increases in efficiency are very important and sufficient computing capabilities are available, it may be worthwhile to use DE even though it is relatively slow to converge.

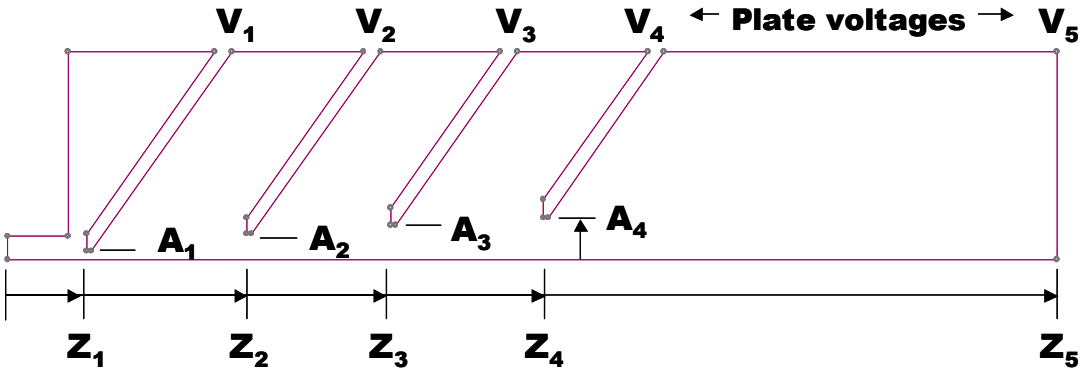

Fig. 3. Idealized collector model studied with optimization system. Parameters that were available for optimization included the plate voltages (V_1-V_5), the plate apertures (A_1-A_4), and the plate axial positions (Z_1-Z_5).

[7] Xiaoling Zhai, personal communication.

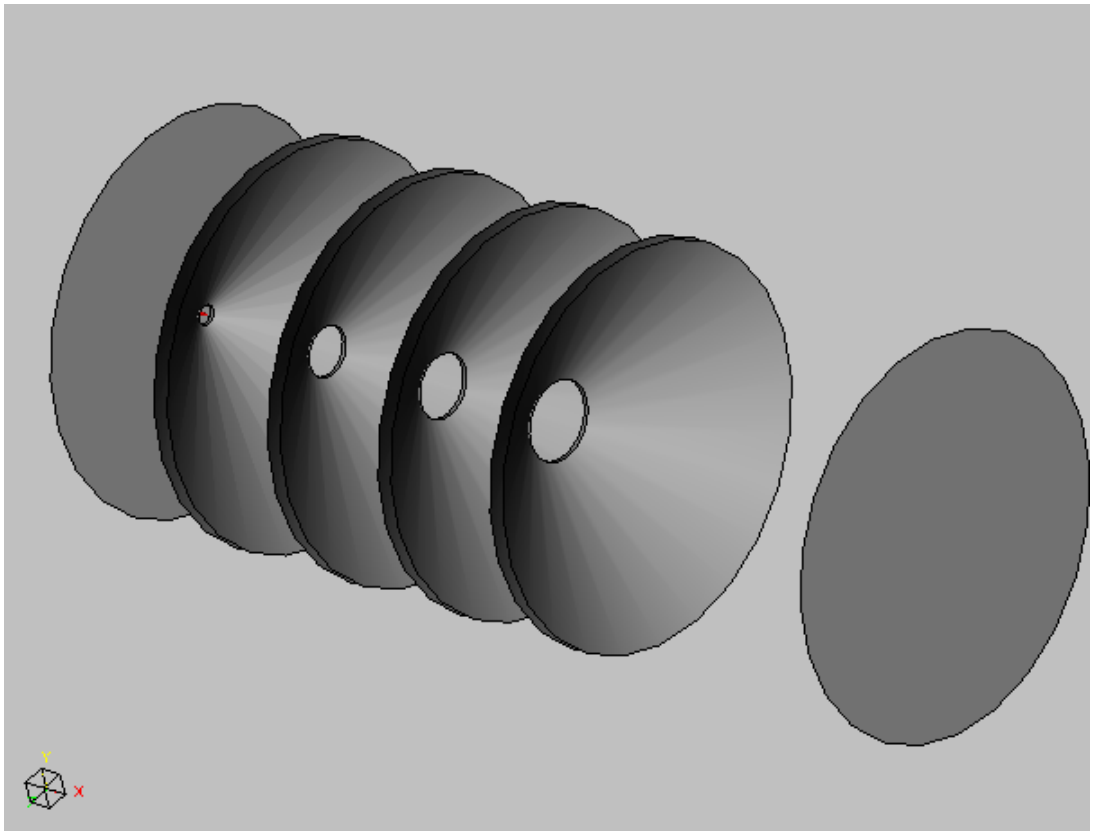

Fig. 4. Image of three-dimensional collector plates created with the initial values of the parameters.

Table 1. Parameter values and efficiencies for optimization of the collector geometry shown in Fig. 1. The initial parameter values and corresponding efficiency are shown in column 1. The columns headed by "Stage n" show the results of the staged optimization where each stage was separately optimized. The final column shows the result when all parameters are used at once in a single optimization.

Step	Initial	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	All
Efficiency	**68.00%**	**74.60%**	**75.60%**	**77.80%**	**79.48%**	**80.13%**	**82.16%**
V1	-2327	-2508	-2508	-2508	-2508	-2508	-2492
V2	-3421	-3421	-3387	-3387	-3387	-3387	-3385
V3	-3987	-3987	-3987	-4377	-4377	-4377	-4385
V4	-4998	-4998	-4998	-4998	-5494	-5494	-5430
V5	-5750	-5750	-5750	-5750	-5750	-5594	-5646
Z1	0.1715	0.1544	0.1544	0.1544	0.1544	0.1544	0.1710
Z2	0.7339	0.7339	0.7173	0.7173	0.7173	0.7173	0.6938
Z3	1.0176	1.0176	1.0176	1.0902	1.0902	1.0902	1.0202
Z4	1.3244	1.3244	1.3244	1.3244	1.3773	1.3773	1.3550
Z5	2.2756	2.2756	2.2756	2.2756	2.2756	2.2323	2.3904
A1	0.0352	0.0491	0.0491	0.0491	0.0491	0.0491	0.0498
A2	0.1078	0.1078	0.1153	0.1153	0.1153	0.1153	0.1055
A3	0.1460	0.1460	0.1460	0.1470	0.1470	0.1470	0.1358
A4	0.1812	0.1812	0.1812	0.1812	0.1655	0.1655	0.1641
# of runs		57	46	39	36	39	261

Secondaries

Several analyses have been made involving multiple generations of secondary particles. The addition of secondaries increases analysis time as expected. Table 2 illustrates the effect of varying the number of secondary generations on efficiency and analysis time. These runs were all for 10 cycles, linear interpolation on an 11K tetrahedral mesh, and with approximately 3000 input particles. An image of the particle distribution for the three-generation case is shown in Fig. 5. As expected, additional secondary generations reduce the efficiency of the collector. Optimization runs involving secondaries are planned for the near future.

Table 2. The effect of adding secondaries to the analysis for the initial parameter vector (shown in column 1 of Table 1).

Number of Secondary Generations	Change in Computed Efficiency (%)	Analysis Time (sec)
0	0	82
1	-17.03	105
2	-20.06	133
3	-20.78	169

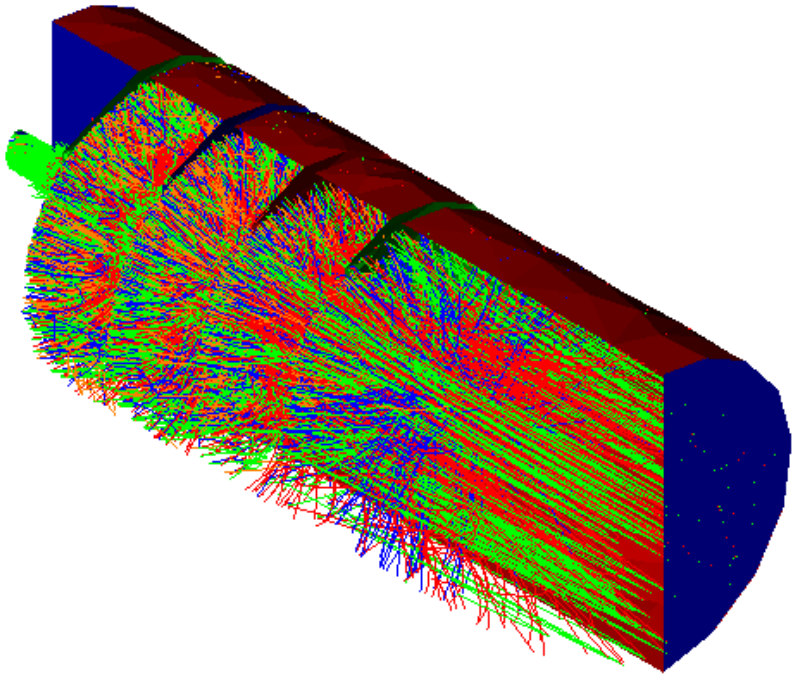

Fig. 5. Image of particles for 3 generations of secondaries.

Adaptive Mesh Refinement (AMR)

A number of AMR analyses have been performed to date. After each solve in an AMR sequence, MICHELLE is responsible for creating a list of elements that should be refined. The MICHELLE solver setup includes parameters for controlling what fraction of the initial elements should be included in the list as well as how these elements should be selected. The following table illustrates the change in efficiency versus AMR iteration number using the following analysis settings:

- 1% refinement fraction.
- Refinement based on charge only.
- 10 cycles.
- 2 generations of secondaries.
- 3000 input particles.
- Cubic interpolation for the potential solve.

From these results it is clear that AMR must be included during optimization for accurate results, particularly if a relatively coarse initial mesh is used. We expect to use AMR in optimizations performed in the Phase 1 Option.

Table 3. The effect of adaptive mesh refinement on the analysis for the initial parameter vector (shown in column 1 of Table 1).

Iteration Number	Number of Elements (x1000)	Change in Efficiency Over Initial Mesh (%)
1	11	-
2	13	+1.23
3	17	+2.35
4	22	+2.74
5	33	+2.82
6	59	+3.20

Plans For Phase 1 Option

Should the Option be funded, we will work in the following areas:

- Demonstration of "realistic" optimization sequence that involve secondaries, adaptive mesh refinement, and repopulated spent beams.
- Work on adaptive mesh refinement process. In particular, to consider the effects of various combinations of refinement metrics on the convergence behavior of AMR as applied to collectors.
- Further work on the interface to improve usability.